



The ACCT Initial Architecture

Project Number: FP6-2005-IST-026996

Deliverable id: D3.1

Deliverable name: The ACCT Initial Architecture

Date: 21.11.2006

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	ACGT
Project Full Name:	Advancing Clinico-Genomic Clinical Trials on Cancer: Open Grid Services for improving Medical Knowledge Discovery
Document id:	D 3.1
Document name:	The ACGT Initial Architecture
Document type (PU, INT, RE):	RE
Version:	DRAFT 1.3
Date:	21.11.2006
Authors:	Juliusz Pukacki
Organization:	PSNC
Address:	Noskowskiego 12/14,Poznan, Poland

Document type PU = public, INT = internal, RE = restricted

<p>ABSTRACT:</p> <p>This deliverable presents the first iteration of defining the architecture for ACGT environment. In first part we are trying to find projects with similar goals to ACGT or using similar technologies to for building software infrastructure. Presented architectures should be treated as a point of reference for our work on defining system architecture</p> <p>Second part is devoted to ACGT architecture itself. It is the first iteration - initial phase - of design process. We are trying to detect and point out the key components and sketch it in general words. There is also simple scenario that could be helpful in understanding the idea behind presented architecture.</p> <p>KEYWORD LIST: Architecture analysis, Grid environment, Biomedical Grid</p>
--

MODIFICATION CONTROL			
Version	Date	Status	Author
1.0	17.11.2006	Draft	Juliusz Pukacki
1.1	20.11.2006	Draft	Juliusz Pukacki
1.3	22.11.2006	Draft	Juliusz Pukacki
1.4	19.12.2006	Draft	Juliusz Pukacki
1.5	12.07.2007	Draft	Juliusz Pukacki

List of contributors:

- Jarek Nabrzyski, PSNC
 - Alberto Anguita, UPM
 - Luis Martín, UPM
 - Dennis Wegener FhG IAIS
-

Table of Contents

1. Executive summary	6
2. Reference Architectures	8
GridLab Architecture	8
InteliGrid Architecture	9
DataMiningGrid	11
caGrid Architecture	13
myGrid Architecture	15
3. ACGT Initial Architecture	18
Hardware Layer	19
Common Grid Infrastructure	19
Advanced Grid Middleware	20
ACGT Business Processes Services	21
User Access and High Level Interoperability Layer	21
4. Layered Architecture Consequences	23
5. Components	25
Common Grid Infrastructure	25
Advanced Grid Middleware	30
ACGT Business Processes Services	38
User Access Layer	41
6. ACGT System Vision - Example	42
References	44

List of Figures

2.1. GridLab architecture	8
2.2. InteliGrid architecture	10
2.3. DataMiningGrid Architecture	12
2.4. caGrid architecture	14
2.5. myGrid architecture	16
3.1. ACGT Architecture - overview	19
4.1. Architecture modification - first option	23
4.2. Architecture modification - second option	24
5.1. ACGT components in the context of architecture	25
5.2. Role of the Mediator	39
6.1. Simple scenario	42

1. Executive summary

ACGT is an Integrated Project (IP) funded in the 6th Framework Program of the European Commission under the Action Line “Integrated biomedical information for better health”. The high level objective of the Action Line is the development of methods and systems for improved medical knowledge discovery and understanding through integration of biomedical information (e.g. using modelling, visualization, data mining and grid technologies). Biomedical data and information to be considered include not only clinical information relating to tissues, organs or personal health-related information but also information at the level of molecules and cells, such as that acquired from genomics and proteomics research.

ACGT focuses on the domain of Cancer research, and its ultimate objective is the design, development and validation of an integrated Grid enabled technological platform in support of post-genomic, multi-centric Clinical Trials on Cancer. The driving motivation behind the project is our committed belief that the breadth and depth of information already available in the research community at large, present an enormous opportunity for improving our ability to reduce mortality from cancer, improve therapies and meet the demanding individualization of care needs.

The first part of deliverable is devoted to describing reference architectures of systems developed in Grid projects or projects for biomedical research:

- GridLab: project from the computational Grids area, focused on designing API for the application developers providing seamless access to the underlying Grid infrastructure. Very important output of GridLab is also set of middleware services for building complex Grid environments.
- IntelliGrid: Grid oriented project introducing the Grid technology to engineering applications area. The goal of IntelliGrid project is to provide a grid-based integration and interoperability infrastructure to complex industries such as construction, automotive and aerospace.
- Data Mining Grid: the idea of the project was to develop tools and services for deploying data mining applications on the grid.
- caGrid: the Cancer Biomedical Informatics Grid, a National Cancer Institute (USA) initiative to link cancer researchers and their data. It is a voluntary virtual informatics infrastructure that connects data, research tools, scientists, and organizations to leverage their combined strengths and expertise in an open environment with common standards and shared tools. One of the primary goals of caBIG is to create a grid infrastructure that supports semantic interoperability. This will enable data and analytical tools provided by the participating cancer research centers and public and private participants to be shared in a manner that is federated and scalable.
- myGrid: myGrid is a UK e-Science project funded by the EPSRC involving five UK universities, the European Bioinformatics Institute and many industrial collaborators. The myGrid project aims to exploit the growing interest in Grid technology, with an emphasis on the Information Grid, and provide middleware layers that make it appropriate for the immediate

needs of bioinformatics.

In the second part of this deliverable initial ACGT architecture is proposed. The whole system is divided into four layers:

- **Hardware Layer:** consists of physical resources - computational nodes, databases, network connections
- **Common Grid Infrastructure:** basic grid components residing on individual resources, providing common grid infrastructure layer for ACGT environment.
- **Advanced Grid Middleware:** more advanced, collective services operating on common grid infrastructure and providing more advanced mechanisms (metascheduling, data management).
- **ACGT Business Processes Services:** higher level services - closer to end user, hiding completely underlying Grid infrastructure, using metadescription for communication with end user and to each other.
- **User Access and High Level Interoperability Layer:** variety of end user application used to exploit functionality provided by the system.

This document provides the basis for the on-going prototyping. Based on the experiences with the prototype, the final version of the architecture will be defined in D3.2.

2. Reference Architectures

In this section five reference architectures are described. They are relevant to the ACGT project from different perspectives. Three of them are architectures originate from the grid oriented projects (GridLab, InteliGrid, Data Minig Grid) and two others are tightly connected to bioinformatics area (caGrid, myGrid).

GridLab Architecture

The main goal of the GridLab Project was to provide a software environment for Grid-enabling scientific applications. To provide an API through which applications access and use available resources. This API directly reflects application needs. Among the intended functionality is the exploration of available resources (CPU, storage, visualization, etc.); remote data access; application migration; etc. The API is concentrated in the Grid Application Toolkit (GAT). The functionality behind the API will be provided by interchangeable capability providers, which may be GridLab services or third-party services.

General view of the GridLab architecture is presented on Figure 2.1, “GridLab architecture”

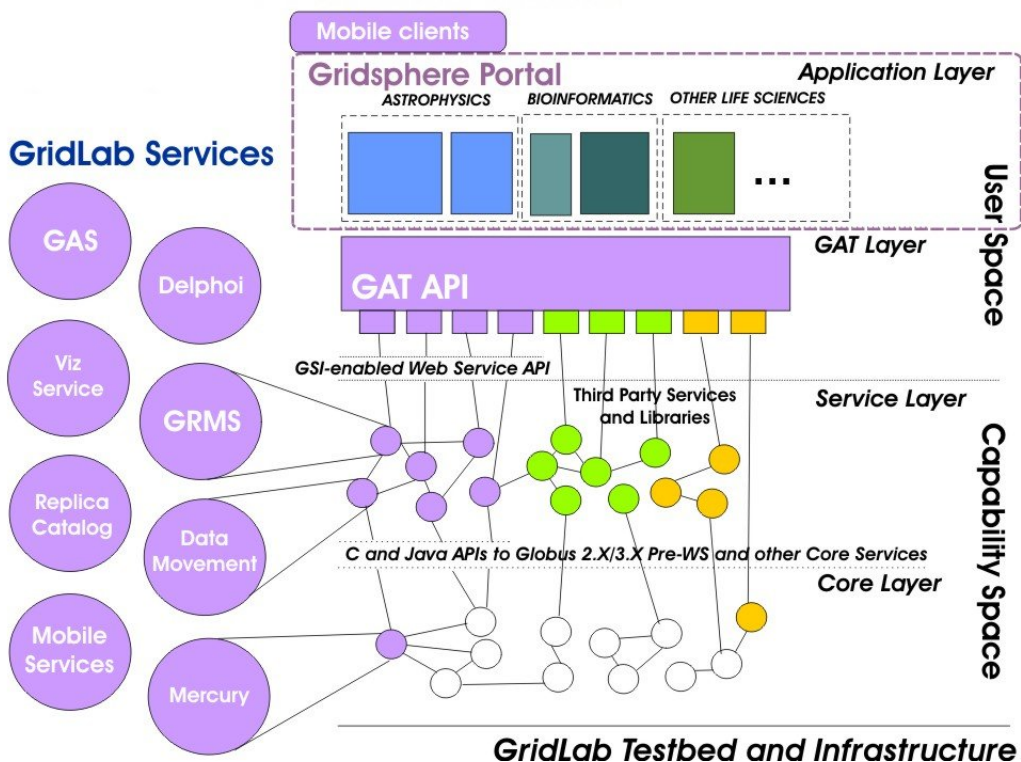


Figure 2.1. GridLab architecture

The applications, located on the highest level of the user space, can access all capability providers they need via the Grid Application Toolkit (GAT) API. The GAT also resides in user space, providing interfaces to the capability providers in the capability space. The GAT main feature is the ability to access various kinds of capability providers via adaptors specific to their communication mechanism, and the semantics of invoking specific capability providers. The GridLab project distinguishes between the GridLab services and third-party services (e.g. low-level Grid services like GIS or GRAM, system services, libraries). Following the requirement, the GAT is able and allowed to access all types of capability providers, on all levels.

Below the User Space on architecture picture, there is the Capability Space. It is divided in two layers: the Service Layer and the Core Layer. The Service Layer consists of collective services developed mainly within the GridLab project. They were implemented using web services technology with GSI security infrastructure. The services on this layer are responsible for: resource management on grid level, data management (data storing, replication, file movement), grid monitoring, grid level security (services authorization). The common feature of the services is that they operate on many other services from the same layer or layer below.

The Core Layer is built with basic services located on individual hardware resources and providing remote access to it. A good example of such a basic service is GRAM component from Globus Toolkit used as a remote interface to computational resources (local queueing system). The other components located in this layer are: hardware monitoring services, resource information providers, data transfer servers, etc.

After the GridLab project had finished, most of the outcome of it was taken to Gridge Toolkit supported and developed by Poznan Supercomputing and Networking Center. Gridge Toolkit is considered to be set of integrated middleware services used to build grid environments. It is one of the most important candidates for building grid infrastructure in ACGT.

InteliGrid Architecture

The goal of InteliGrid is to provide a grid-based integration and interoperability infrastructure to complex industries such as construction, automotive and aerospace. Projects vision of future engineering is a flexible, secure, robust, ambient accessible, interoperable, pay-per demand access to information, communication and processing resources.

One of the major technological goals of InteliGrid is to make the grid infrastructure available to the small to medium enterprise (SME) companies that are providing and using engineering software and services. The core competencies of these companies are in topics such as structural mechanics or 3D solid modelling and not in the latest trends in middleware technology or interoperability platforms. The project helps SMEs to enhance their applications with grid-computing capabilities and increase efficiency provided by semantics.

InteliGrid created a platform supporting various emerging international standards (WSRF, OGSA, WS-I, etc.) for the semantic integration of distributed engineering services using intelligent ontology services to bridge the gap between technical grid concepts and engineering domain concepts. Key building blocks of this solution are the InteliGrid Portal, InteliGrid Document Management, InteliGrid Ontology Services, InteliGrid Business Object Services as well as sever-

al semantically annotated applications.

Figure 2.2, “InteliGrid architecture” presents the architecture of the system created in InteliGrid project.

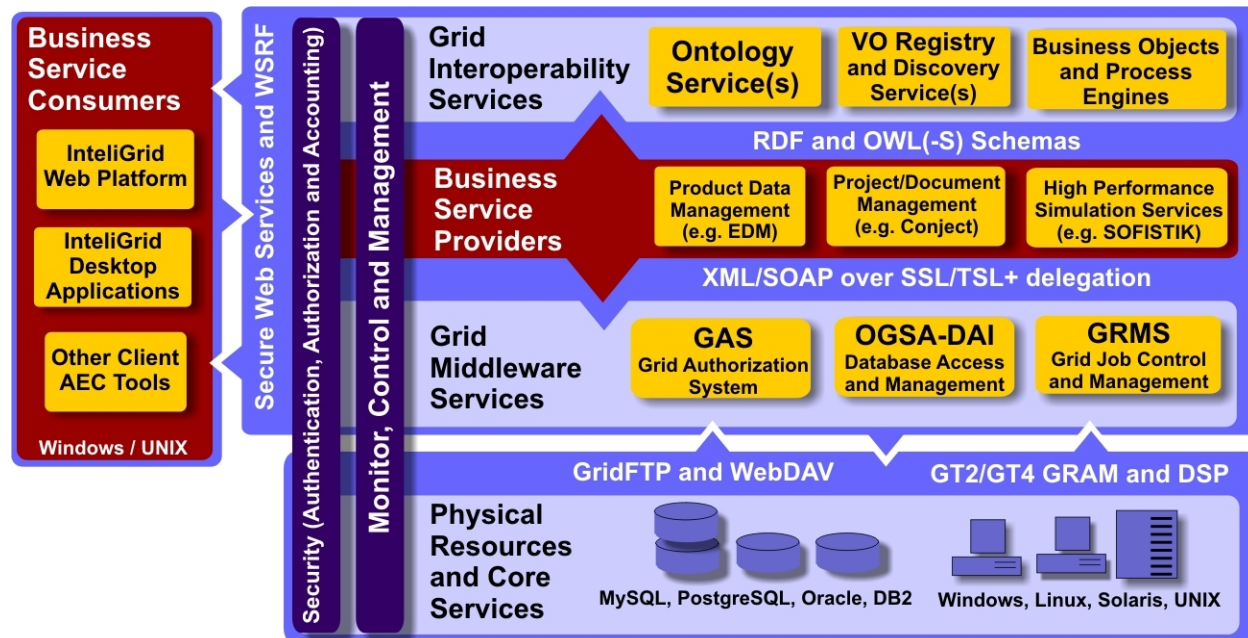


Figure 2.2. InteliGrid architecture

The low layer of the InteliGrid architecture consists of various physical infrastructure resources (both data/storage and computing resources) offered by suppliers to InteliGrid Platform. All these resources are available through well defined secure protocols and grid-enabled generic core services for remote data access: GridFTP and WebDAV, and also for remote application submission and control: GT2/GT4 GRAM and DSP. All core services can be accessed via a network through well defined interfaces and communication protocols to provide functionality to requesters.

Requesters to core services are located on a higher-layer called the InteliGrid middleware, or InteliGrid Platform. The InteliGrid Platform consists of generic Grid Middleware Services such as Authorization Service (GAS), OGSA-DAI and Grid metascheduler (GRMS) based on mature grid technologies and their open source reference implementations. There are also complementary Grid Interoperability Services including Ontology Service(s), VO Registry and Discovery Service(s) and Business Objects/Process Engines based on semantics and knowledge management technologies.

Business Services as a part of InteliGrid Platform are available for end users (Business Service Consumers) on-demand when they need them. In general, business services provide secure SOAP-based interfaces, based on Web Services and WSRF technologies, so various clients running on local machines are able to discover their location, policies, interfaces and protocols. Business services based on Web Services and WSRF technologies are in fact distributed soft-

ware components that provide information to applications rather than to humans through application-oriented interfaces.

DataMiningGrid

The DataMiningGrid project was a shared cost Strategic Targeted Research Project (STREP) granted by the European Commission (grant no. IST-2004-004475). It was part of the Sixth Framework Programme of the Information Society Technologies Programme. The idea of the project was to develop tools and services for deploying data mining applications on the grid by meeting the requirements of modern distributed data mining scenarios. Within the project new ways to utilize grid computing technology for applying data mining, especially regarding integration of applications, their execution in the grid, and user-friendliness were found. The DataMiningGrid system has been designed to meet the requirements of distributed data mining scenarios. Based on the grid middleware Globus Toolkit and other open technology and standards, the DataMiningGrid system provides tools and services facilitating the grid-enabling of data mining applications without major intervention in the application. In order to address a wide range of requirements arising from the need and context to mine data in distributed computing environments, demonstration and testing of the DataMiningGrid system was done on the basis of a selected set of real-world applications which represent use cases from a diverse set of sectors:

- Automotive industry (customer relationship management, which uses complex text-mining and ontology learning grid-enabled applications)
- Health sector (grid-enabled and privacy-preserving analysis of geographically widely distributed medical patient records)
- Systems biology (grid-assisted re-engineering of gene regulatory networks and analysis of proteins using computational simulations)
- Ecological modelling (produce more accurate ecological models much faster)
- Text mining (text classification, ontology learning and analysis of digital libraries)

The underlying approach of the DataMiningGrid project is depicted in Figure 2.3, "DataMiningGrid Architecture"

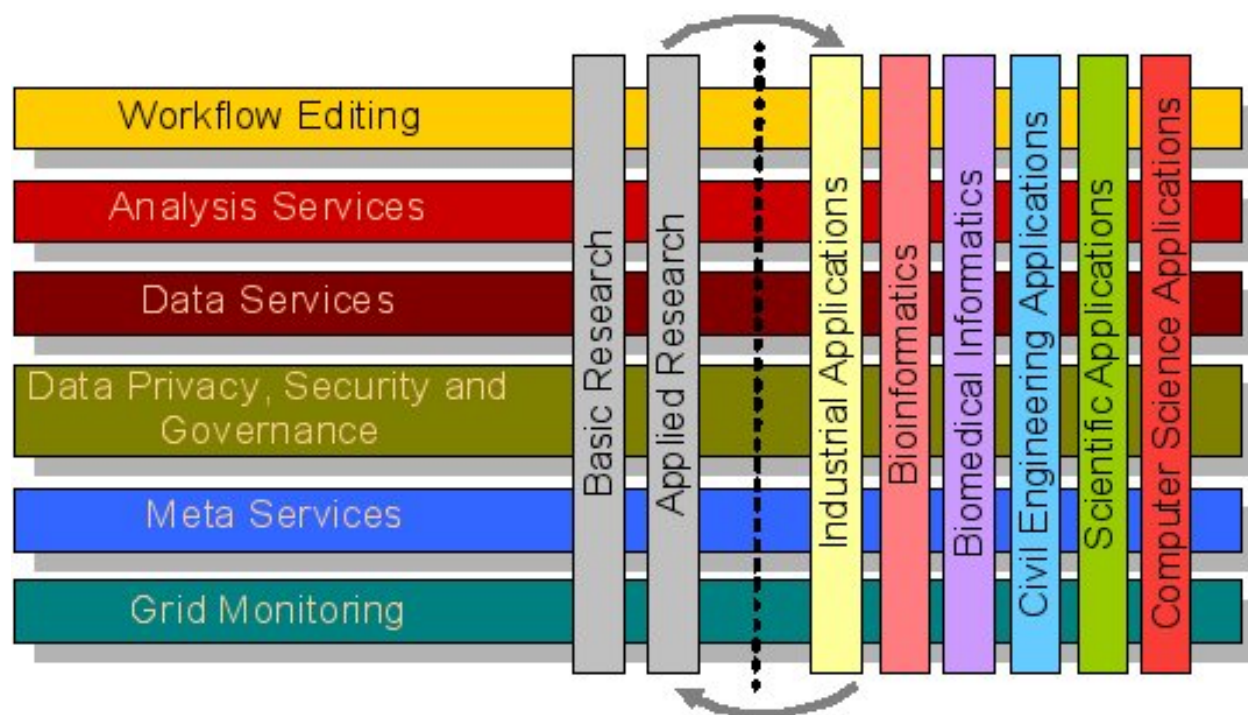


Figure 2.3. DataMiningGrid Architecture

The philosophy in the DataMiningGrid project was to avoid implementing specialized services and client-side components for each data mining application which is integrated in the grid environment. Instead an XML-based meta-data schema, the DataMiningGrid Application Description Schema (ADS), has been developed, which allows application developers to describe each single algorithm in detail regarding its options, inputs, outputs, resource requirements and additional information (vendor, version, etc.). The application descriptions are stored in a searchable repository, which can be queried at any time to provide an up-to-date view of the applications currently available in the grid. Furthermore, the descriptions can be used by generic client-side components (e.g. a workflow editor, Web portal) for providing a dynamic GUI for specifying user settings and additional help for specifying input data and resource requirements. The fact that instead of implementing specialized services and client-side components for each data mining application more generic components were developed also enables sophisticated features such as execution of applications on remote machines, selected automatically by a resource broker, without prior installation of these applications and the ability to utilize thousands of machines for running highly parallel applications.

To meet the requirements, the following critical technology components were developed by the project:

- Workflow editor and manager that facilitates the composition, execution and management of complex data mining workflows in grid environments
- Data Mining Application Enabler which is used to grid enable existing data mining applica-

tions

- Grid-enabled data access and integration services that allow the users to identify (locate), access, integrate and interface distributed data sources in a flexible way;
- Information services and a resource broker which will execute data mining applications in grid environment

The DataMiningGrid architecture consists of 4 different layers. The bottom layer contains hardware and software resources. Typical basis hardware is e.g. storage devices, processing element and networks etc. The software is the actual data to mine, algorithms etc. A special kind of resource is a cluster, which is a collection of coupled computers with a local scheduler. The second layer provides the main grid middleware functionality to DataMiningGrid system. The DataMiningGrid system bases on GT4, which meets the requirements of OGSA and WSRF. On top of this layer there is a layer containing higher level services (resource broker, information services, data services). The DataMiningGrid resource broker, based on the GridBus resource broker, is responsible for resource discovery and section, job preparation and submission and job monitoring. The InformationServices are necessary for providing information on the resources and the applications available in the grid. The DataMiningGrid DataServices are responsible for data access e.g. by providing OGSA-DAI services for performing data mining related operations. The top layer is the client layer which depicts all client side components. The DataMiningGrid system comprises two different types of clients, a workflow editor and a web-based client. The DataMiningGrid workflow editor and manager bases on Triana and contains client side components which address users which have knowledge about data mining and workflow composing but no knowledge about the underlying grid technology. For even less sophisticated users the web-based clients provide the simplest kind of user interface.

caGrid Architecture

The caGRID project aims to produce a software system capable of handling biological data that crosses compartments. The approach is to leverage existing technologies for the standardization of data representation and semantics and couple them to a common data integration architecture. caCORE is NCICB's platform for data management and semantic integration, built using formal techniques from the software engineering and computer science communities. caCORE defines a data model specified using industry standard techniques to define common biological constructs (objects). By mapping compartments to these common data objects, the process of data integration can commence. More importantly, activation energy needed to cross and interoperate amongst compartments is lowered enough to allow for more sophisticated biological discovery.

The overall architecture of caGrid is presented on Figure 2.4, “caGrid architecture”.

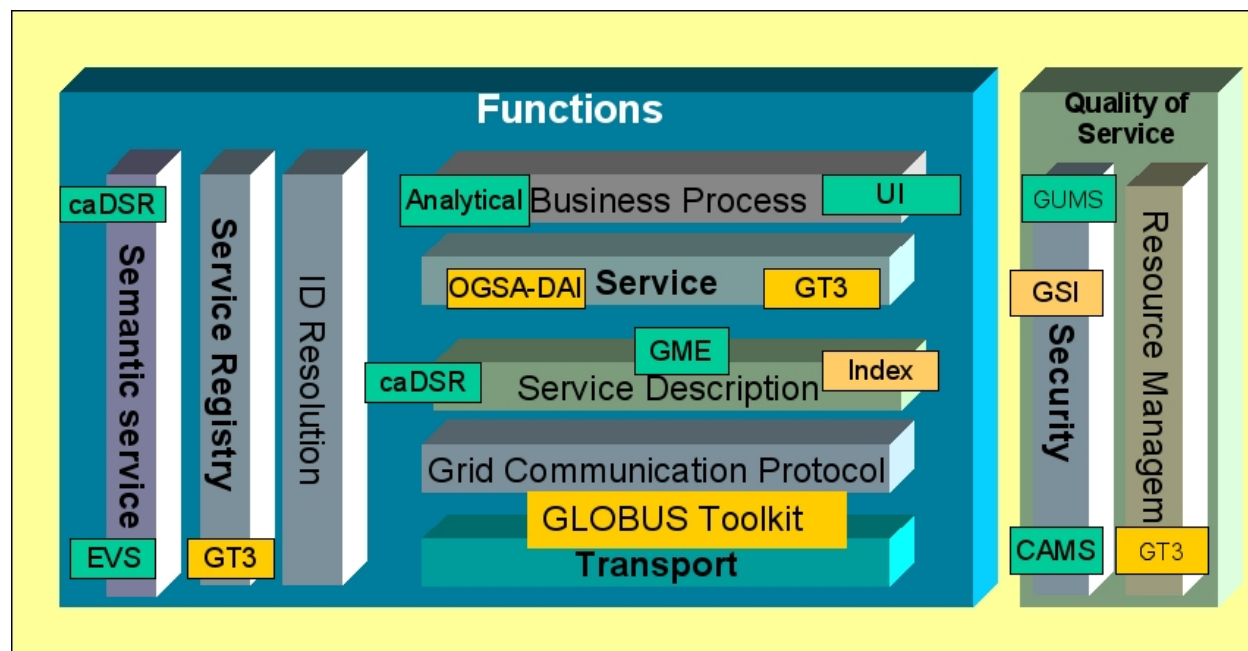


Figure 2.4. caGrid architecture

As a primary principle of caBIG is open standards, caGrid is built upon the community-driven standards of Open Grid Services Architecture (OGSA). OGSA is the current specification for grid infrastructure and has been developed over the past several years in the Global Grid Forum (GGF), the community of users, developers, and vendors leading the global standardization effort for grid computing. The current release of caGrid is built using the Globus Toolkit 3.2 (GT3.2) and the OGSA Data Access Integration (OGSA-DAI) framework. GT3.2 is a reference implementation of OGSA for service creation, deployment, and invocation and provides core services such as the Index Service for service registration and discovery and Globus Security Infrastructure (GSI) for security. The OGSA-DAI framework is an implementation of OGSA for data services. It provides a set of interfaces and runtime support for implementing and deploying data sources as Grid services.

Extending beyond the basic grid infrastructure, caBIG specializes these technologies to better support the needs of the cancer research community. A primary distinction between basic grid infrastructure and the requirements identified and implemented in caGrid is the attention given to data modeling and semantics. caBIG adopts a model-driven architecture best practice and requires that all data types used on the grid are formally described, curated, and semantically harmonized. These efforts result in the identification of common data elements, controlled vocabularies, and object-based abstractions for all cancer research domains. caGrid leverages existing NCI data modeling infrastructure to manage, curate, and employ these data models. Data types are defined in caCORE UML and converted into ISO/IEC 11179 Administered Components, which are in turn registered in the Cancer Data Standards Repository (caDSR). The definitions draw from vocabulary registered in the Enterprise Vocabulary Services (EVS), and their relationships are thus semantically described.

In caGrid, both the client and service APIs are object oriented, and operate over well-defined and curated data types. Clients and services communicate through the grid using Globus grid clients and service infrastructure, respectively. The grid communication protocol is XML based, and thus the client and service APIs must transform the transferred objects to and from XML. This XML serialization of caGrid objects is restricted, as each object that travels on the grid must do so as XML which adheres to an XML schema registered in the Global Model Exchange (GME). As the caDSR and EVS define the properties, relationships, and semantics of caBIG data types, the GME defines the syntax of the XML serialization of them. Furthermore, Globus services are defined by the Web Service Description Language (WSDL). The WSDL describes the various operations the service provides to the grid. The inputs and outputs of the operations, among other things, in WSDL are defined by XML schemas. As caBIG requires that the inputs and outputs of service operations use only registered objects, these input and output data types are defined by the XSDs which are registered in GME. In this way, the XSDs are used both to describe the contract of the service and to validate the XML serialization of the objects which it uses.

As caBIG aims to connect data and tools from 50+ disparate cancer centers, a critical requirement of its infrastructure is that it supports the ability of researchers to discover these resources. caGrid enables this ability by taking advantage of the rich structural and semantic descriptions of data models and services that are available. Each service is required to describe itself using caGrid standard service metadata. When a grid service is connected to the caBIG grid, it registers its availability and service metadata with a central indexing registry service (Index Service). This service can be thought of as the “yellow pages” and “white pages” of caBIG. A researcher can then discover services of interest by looking them up in this registry. caGrid 0.5 provides a series of high-level APIs and user applications for performing this lookup which greatly facilitate the process.

myGrid Architecture

myGrid is an e-Science research project developing open source high-level middleware to support in-silico experiments in biology. In-silico experiments use databases and computational analysis rather than laboratory investigations to test hypothesis. In silico experiments are procedures using computer based information repositories and computational analysis adopted for testing hypothesis or to demonstrate known facts. myGrid focuses on data intensive experiments that combine use of applications and database queries. The user is helped to create workflows (a.k.a. experiments), sharing and discovering others’ workflows and interacting with the workflows as they run.

The project’s goal is to provide middleware services as a toolkit to be adopted and used in a “pick and mix” way by bioinformaticians, tool builders and service providers who in turn produce the end applications for biologists. The target environment is open, by which it means that services and their users are decoupled. Services are not just used solely by their publishers but by users unknown to the service provider, who may use them in unexpected ways.

The my Grid middleware framework employs a service-based architecture, firstly prototyped with Web Services but with an anticipated migration path to the Open Grid Services Architecture (OGSA) gives an account of the conversion of two my Grid services to OGSi services. The middleware services are intended to be collectively or selectively adopted by bioinformaticians, tool

builders and service providers who in turn produce the end applications for biologists. Figure 2.5, “myGrid architecture” shows the layered middleware stack of services.

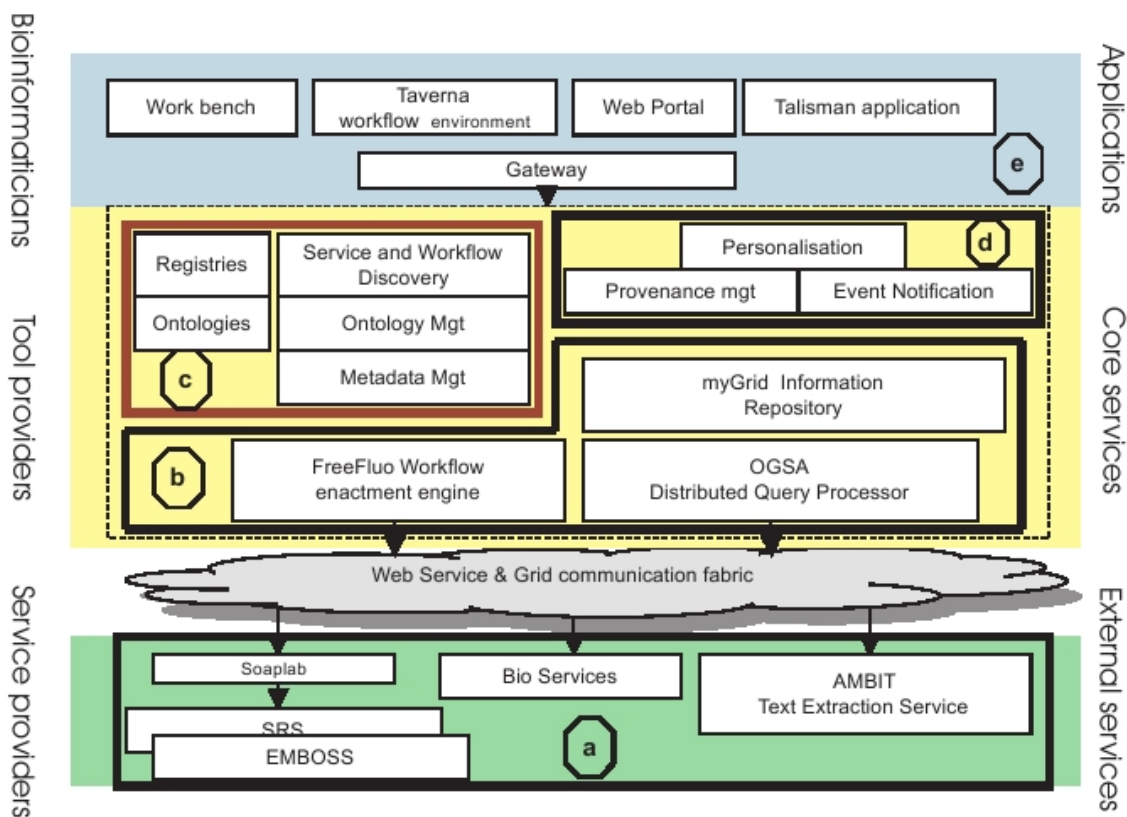


Figure 2.5. myGrid architecture

The primary services to support routine in silico experiments fall into four categories:

- services that are the tools that will constitute the experiments, that is: specialised services such as AMBIT text extraction, and external third party services such as databases, computational analysis, simulations etc, wrapped as web services by Soaplab if required;
- services for forming and executing experiments, that is: workflow management services, information management services, and distributed database query processing;
- semantic services for discovering services and workflows, and managing metadata, such as: third party service registries and federated personalised views over those registries, ontologies and ontology management;
- services for supporting the e-Science scientific method and best practice found at the bench but often neglected at the workstation, specifically: provenance management and change no-

tification.

The final layer (e) constitutes the applications and application services that use some or all of the services described above.

3. ACGT Initial Architecture

Layering is one of the most popular architectural design pattern. It structures software, so it can be decomposed into groups of subtasks such that each group of subtasks is at a particular level of abstraction. It also helps to split responsibilities of the different groups of components in the system. At first layered architecture were used to divide application presentation features from application logic and database access (three-tier model). The highest level layer in that model is used as an interface for the end user that translates tasks and results to something the user can understand. Logic Layer coordinates the application, process commands, makes logical decisions. It also moves and processes data between two surrounding layers. Within Data Layer all the information required by the application is stored and retrieved from storage databases or filesystem.

Such a classical approach as described above can be used as a reference point for more complex applications and systems. It is possible to extend basic model by splitting particular layers to additional ones depending on the characteristics of the concrete problem. The actual meaning of the layers also can be result of modelled environment.

Nowadays, one of the most important trend in system design is Service Oriented Architecture (SOA). It defines software infrastructure as a loosely coupled software services to support the requirements of the business processes and software users. In an SOA environment, resources on a network are made available as independent services that can be accessed without knowledge of their underlying platform implementation. A service-oriented architecture is not tied to a specific technology. It may be implemented using a wide range of interoperability standards but the most important standard in the context of SOA is Web Services. The key issue is that all services that building some environment are independent from each other, they just publish their interfaces and that is enough for any other entity in the system to invoke methods means 'use' the service.

Architecture proposed for the ACGT system is based on two described briefly in previous paragraphs approaches. Components of the system are splinted into four abstract layers to group functionality depending on 'distance' from physical resources, and level of abstractions they operate on. The proposed layers are (Figure 3.1, "ACGT Architecture - overview"):

- Hardware Layer
- Common Grid Infrastructure Layer
- Advanced Grid Middleware
- ACGT Business Process (Model) Services
- User Access Layer

Comparing to classical three-tier model there are two additional layer that are very important from Grid technology perspective.

Despite the fact that services are placed in some abstraction layers, within one layer (and also from neighbouring layers) they are building kind of service-oriented environment. So it is possible to present those services in a context of the Service Oriented Architecture. As the SOA requires the services are loosely coupled, very well defined, by their interfaces descriptions, can be managed separately. The very important feature of the SOA is a dynamism - the services can be added (and removed) to the environment in dynamic manner. On the other hand however it is necessary to define initial set of services based on user requirements analysis.

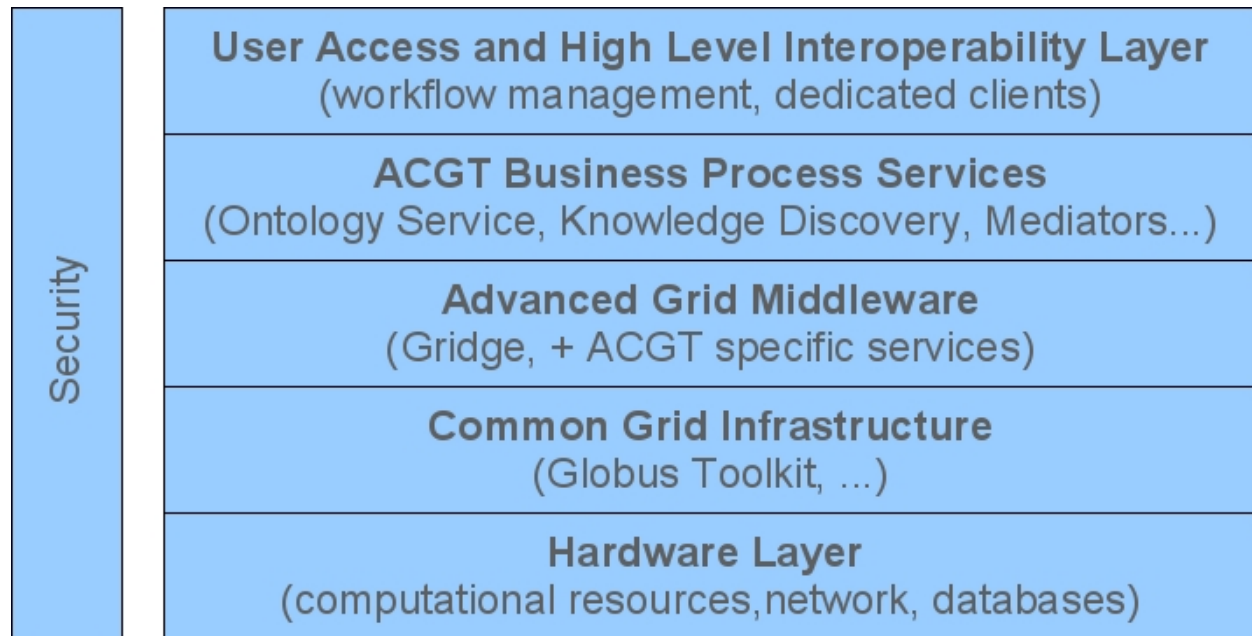


Figure 3.1. ACGT Architecture - overview

Hardware Layer

It is the lowest level layer in the ACGT system architecture. It consists of physical resources scattered throughout the Europe, connected using some network infrastructure. There are two main types of resources: computational nodes and data nodes: storage system or databases.

ACGT activity which affect this layer will be reduced to administration of machines which are the part of the testbed. There can be also some work required in case of creation and support of new database.

Common Grid Infrastructure

All the resources from Hardware Layer should be accessed remotely in as much unified way as possible. Common Grid Infrastructure is responsible for providing mechanisms for remote access to resources in a "grid" way.

Components from that layer located (installed) on individual resources and provides remote access to it (and only it) in a context of concrete functionality. This layer should provide following functionalities:

- job execution an control
- file transfer
- data bases access
- monitoring sensors
- resource information providers

For ACGT purposes, already existing technologies can be used. The most probable solution is Globus Toolkit which comprises all required components for building common grid infrastructure. Some additional development can be done in context of access to specific data sources.

Most of the scenarios in ACGT are focused on date integration, so remote access to distributed data sources is the most important technological challenge.

Advanced Grid Middleware

Advanced Grid Middleware is a layer responsible for providing more advanced mechanisms in the Grid environment. Services from this layer can be described as "collective" because they operate on set of lower level services, to realize more advanced actions - eg. metascheduling service that submits jobs to different local queuing systems.

The same way as previous one, this layer will be built using existing software. Solution which is considered is Gridce Toolkit, that provide following functionality:

- metascheduling
- data management (on file level)
- services authorization
- grid monitoring
- mobile user support

Some additional components will be required for distributed access to heterogeneous data sources (databases) - OGSA-DAI technology is considered.

Adding new components to this layer there should be a requirement or recommendation that it

should be implemented in compliant technology - web services.

ACGT Business Processes Services

This layer consists of components which are not aware of physical resources and Grid environment. Grid is used as a whole, to perform more abstract actions, and to get information required by end user. Services of this layer are using abstract description of world defined as some ontology. There are also services that are able to translate high level description to language understandable by lower layer (Grid) . It provides higher level integration of different resources and data and makes them more similar to real word objects.

It is possible do define following functionality for Business Processes Services:

- ontology description provisioning
- execution and control of workflows
- translation of abstract object to resources names
- translation of abstract actions to grid calls
- virtual organization management
- knowledge discovery
- biomedical algorithms

Services of this layer are specific for ACGT environment, so it will not be possible to use existing ones very often. Most services of this layer will be developed by ACGT and most of the results of development process will land on this layer.

User Access and High Level Interoperability Layer

The layer gathers all application and tools that provide access to the ACGT Environment for end user. There can be a wide variety of software, developed using different technology:

- portals
- standalone applications
- clients dedicated to specific operation
- workflows editors

- visualization tools

Client applications are in the most cases tightly connected to functionality provided by underlying layer. They are also going to be used by different but specific groups of users. Thus it would be difficult to use general tool but there will be need to create dedicated application. These two arguments are showing that User Access Layer will be mostly developed by ACGT.

4. Layered Architecture Consequences

Using layering as a architectural design pattern for ACGT brings some consequences to the shape of the system. One of the important ones are restrictions in communication between layers - components from layer N can only call components from layer N - 1 (below) and can be accessed by elements from layer N +1 (above). In some cases such restrictions are too strong that why it is possible to weaken it by using two mechanisms:

- Denoting that some layer can access more that one layers below

A good example showing that case is specialized client portal that can communicate directly with grid metascheduler to perform job submission. But we should be aware that by performing such an action it is required to talk in a language understandable directly by metasched-uler - can not use abstractions of Business Process Services.

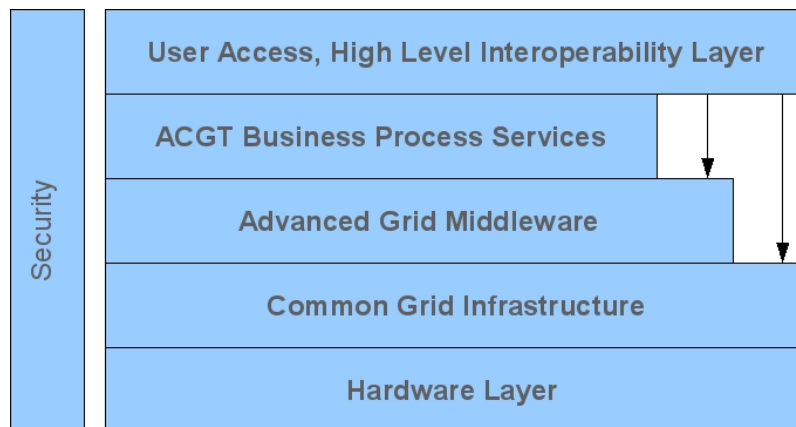


Figure 4.1. Architecture modification - first option

- Adding additional vertical layer that crosses all horizontal layers (e.g. security mechanisms that should be present in all layers)

The example of this case is security infrastructure, that should be consistant and present on each layer. The other possibility is some registry service used for indexing services from different layers

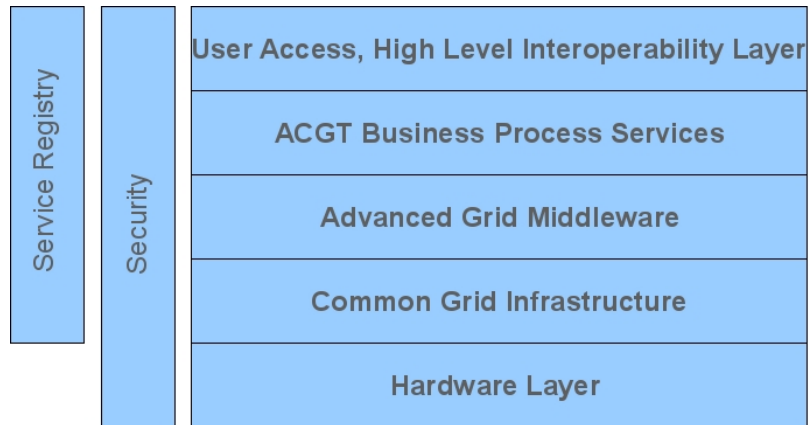


Figure 4.2. Architecture modification - second option

5. Components

During requirements analysis some functional components were identified. Some of them came from end users needs, and the others are important from the point of view of grid infrastructure building. Yellow color on picture below denotes services that need to be developed by ACGT consortium by themselves.

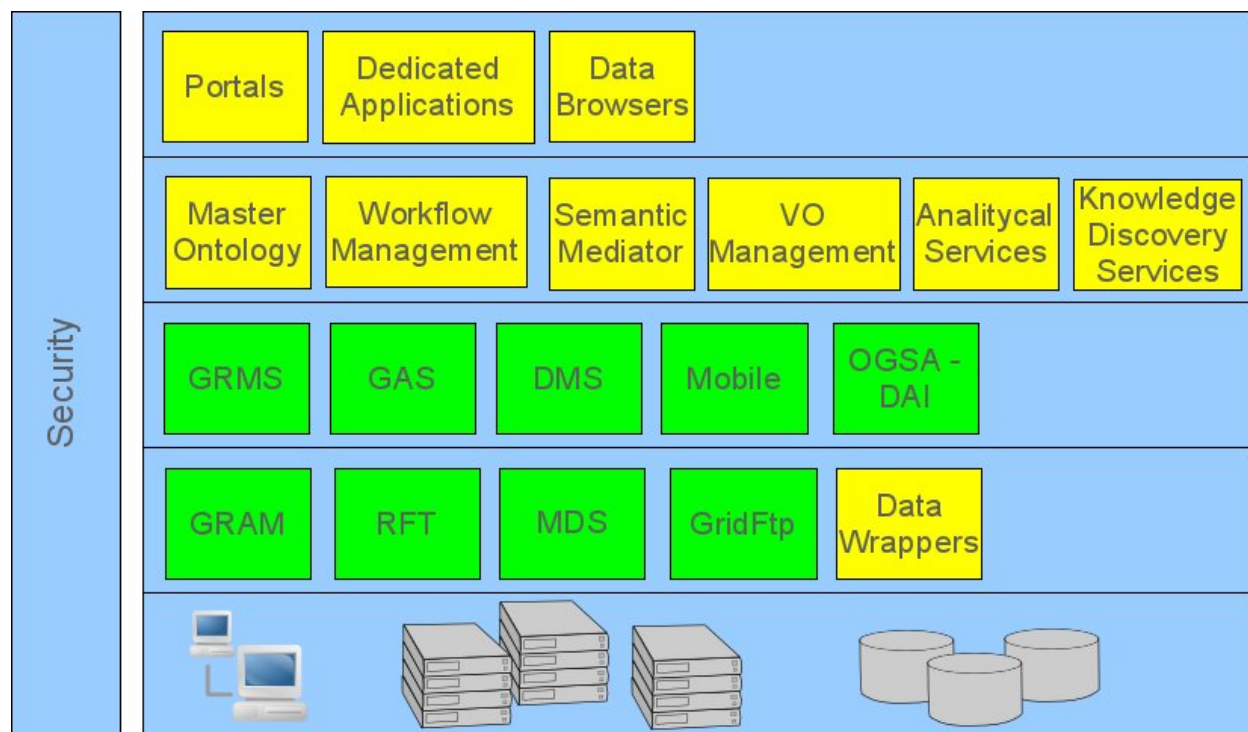


Figure 5.1. ACGT components in the context of architecture

Common Grid Infrastructure

Most of the components of this layer come from Globus Toolkit. The newest version of Globus Toolkit (GT4) is the candidate to be used for the ACGT project. Globus is the well known standard for building Grid environments. The toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection, and portability. It is packaged as a set of components that can be used either independently or together to develop applications.

Following GT4 components are the most important:

- GRAM

Grid Resource Allocation Management (GRAM) is a component of the Globus Toolkit responsible for job execution. Grid computing resources are typically operated under the control of a scheduler which implements allocation and prioritization policies while optimizing the execution of all submitted jobs for efficiency and performance. GRAM is not a resource scheduler, but rather a protocol engine for communicating with a range of different local resource schedulers using a standard message format. Rather than consisting of a monolithic solution, GRAM is based on a component architecture at both the protocol and software implementation levels. This component approach serves as an ideal which shapes the implementation as well as the abstract design and features.

There are two implementation of GRAM service: "web services" and "pre-web services" Unix server suite to submit, monitor, and cancel jobs on Grid computing resources. Both systems are known under the moniker "GRAM", while "WS GRAM" refers only to the web service implementation.

Job management with GRAM makes use of multiple types of service:

- Job management services represent, monitor, and control the overall job life cycle. These services are the job-management specific software provided by the GRAM solution.
- File transfer services support staging of files into and out of compute resources. GRAM makes use of these existing services rather than providing redundant solutions; WS GRAM has further refactored some file transfer mechanisms present in pre-web service GRAM.
- Credential management services are used to control the delegation of rights among distributed elements of the GRAM architecture based on users' application requirements. Again, GRAM makes use of more general infrastructure rather than providing a redundant solution, and WS GRAM has continued this refactoring to better separate credential management at the protocol level.

For WS GRAM, the Globus Toolkit software development environment, and particularly WSRF core, is used to implement distributed communications and service state. For pre-web service GRAM, the "gatekeeper" daemon and GSI library are used for communications and service dispatch.

WS GRAM utilizes WSRF functionality to provide for authentication of job management requests as well as to protect job requests from malicious interference, while pre-web service GRAM uses GSI and secure sockets directly. The use of GRAM does not reduce the ability for system administrators to control access to their computing resources. The use of GRAM also does not reduce the safety of jobs users run on a given computing resource. To protect users from each other, jobs are executed in appropriate local security contexts, e.g. under specific Unix user IDs based on details of the job request and authorization policies. Additionally, GRAM mechanisms used to interact with the local resource are design to minimize the privileges required and to minimize the risks of service malfunction or compromise. A client may delegate some of its rights to GRAM services in order to facilitate the above functions, e.g. rights for GRAM to access data on a remote storage element as part of the job execution.

Additionally, the client may delegate rights for use by the job process itself. With pre-web service GRAM, these two uses of rights are inseparable, while WS GRAM provides separate control for each purpose (while still allowing rights to be shared if that is desired).

WS GRAM provides an "at most once" job submission semantics. A client is able to check for and possibly resubmit jobs, in order to account for transient communication errors without risk of running more than one copy of the job. Similarly, pre-web service GRAM provides a two-phase submission mechanism to submit and then commit a job to be run. While many jobs are allowed to run to their natural completion, GRAM provides a mechanism for clients to cancel (abort) their jobs at any point in the job life cycle.

WS GRAM provides for reliable, high-performance transfers of files between the compute resource and external (gridftp) data storage elements before and after the job execution. Pre-web service GRAM can also stage with gridftp systems but with less flexible reliable-transfer logic driving its requests. GRAM supports a mechanism for incrementally transferring output file contents from the computation resource while the job is running. WS GRAM uses a new mechanism to allow arbitrary numbers of files to be transferred in this fashion, while pre-web service GRAM only supports incremental transfer of the job's standard output and error streams.

- MDS

The Monitoring and Discovery System (MDS) is a suite of web services to monitor and discover resources and services on Grids. This system allows users to discover what resources are considered part of a Virtual Organization (VO) and to monitor those resources. MDS services provide query and subscription interfaces to arbitrarily detailed resource data and a trigger interface that can be configured to take action when pre-configured trouble conditions are met. The services included in the WS MDS implementation (MDS4), provided with the Globus Toolkit 4, acquire their information through an extensible interface which can be used to: query WSRF services for resource property information, execute a program to acquire data, or interface with third-party monitoring systems.

Grid computing resources and services can advertise a large amount of data for many different use cases. MDS4 was specifically designed to address the needs of a Grid monitoring system – one that publishes data that is available to multiple people at multiple sites. As such, it is not an event handling system, like NetLogger, or a cluster monitor on its own, but can interface to more detailed monitoring systems and archives, and can publish summary data using standard interfaces.

MDS4 includes two WSRF-based services: an Index Service, which collects data from various sources and provides a query/subscription interface to that data, and a Trigger Service, which collects data from various sources and can be configured to take action based on that data. An Archive Service, which will provide access to historic data, is planned for a future release. The Index Service is a registry similar to UDDI, but much more flexible. Indexes collect information and publish that information as resource properties. Clients use the standard WSRF resource property query and subscription/notification interfaces to retrieve information from an Index. Indexes can register to each other in a hierarchical fashion in order

to aggregate data at several levels. Indexes are “self-cleaning”; each Index entry has a life-time and will be removed from the Index if it is not refreshed before it expires. Each Globus container that has MDS4 installed will automatically have a default Index Service instance. By default, any GRAM, RFT, or CAS service running in that container will register itself to the container’s default Index Service. The Trigger Service collects information and compares that data against a set of conditions defined in a configuration file. When a condition is met, or triggered, an action takes place, such as emailing a system administrator when the disk space on a server reaches a threshold.

In addition to the services described above, MDS4 includes several additional software components, including an Aggregator Framework, which provides a unified mechanism used by the Index and Trigger services to collect data. The Aggregator Framework is a software framework used to build services that collect and aggregate data. Services (such as the Index and Trigger services) built on the Aggregator Framework are sometimes called aggregator services, and have the following in common:

- They collect information via Aggregator Sources. An Aggregator Source is a Java class that implements an interface (defined as part of the Aggregator Framework) to collect XML-formatted data
- They use a common configuration mechanism to maintain information about which Aggregator Source to use and its associated parameters (which generally specify what data to get, and from where). The Aggregator Framework WSDL defines an [aggregating service group entry type] that holds both configuration information and data. Administrative client programs use standard [WSRF Service Group registration mechanisms] to register these service group entries to the aggregator service.
- They are self-cleaning – each registration has a lifetime; if a registration expires without being refreshed, it and its associated data are removed from the server.

MDS4 includes the following three Aggregator Sources:

- the Query Aggregator Source, which polls a WSRF service for resource property information,
- the Subscription Aggregator Source, which collect data from a WSRF service via WSRF subscription/notification,
- the Execution Aggregator Source, which executes an administrator-supplied program to collect information.

Depending on the implementation, an Aggregator Source may use an external software component (for example, the Execution Aggregator Source uses an executable program), or a WSRF service may use an external component to create and update its resource properties (which may then be registered to an Index or other aggregator service, using the Query or Subscription Aggregator Source). This set of components are referred to as Information Providers. Currently, MDS4 includes the following sources of information: Hawkeye Information Provider, Ganglia Information Provider, WS GRAM, Reliable File Transfer Service

(RFT), Community Authorization Service (CAS), any other WSRF service that publishes resource properties

- RFT

RFT is a Web Services Resource Framework (WSRF) compliant web service that provides "job scheduler"-like functionality for data movement. You simply provide a list of source and destination URLs (including directories or file globs) and then the service writes your job description into a database and then moves the files on your behalf. Once the service has taken your job request, interactions with it are similar to any job scheduler. Service methods are provided for querying the transfer status, or you may use standard WSRF tools (also provided in the Globus Toolkit) to subscribe for notifications of state change events. The provided service implementation is installed in a web services container (like all web services) and a very simple client. There are Java classes available for custom development, but due to lack of time and resources, work is still needed to make this easier.

- GridFTP

GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. The GridFTP protocol is based on FTP, the highly-popular Internet file transfer protocol. A set of protocol features and extensions defined already in IETF RFCs was selected and added a few additional features to meet requirements from current data grid projects.

GridFTP functionality includes some features that are supported by FTP extensions that have already been standardized (RFC 959) but are seldom implemented in current systems. Other features are new extensions to FTP.

- Grid Security Infrastructure and Kerberos support: Robust and flexible authentication, integrity, and confidentiality features are critical when transferring or accessing files. GridFTP must support GSI and Kerberos authentication, with user controlled setting of various levels of data integrity and/or confidentiality. GridFTP implements the authentication mechanisms defined by RFC 2228, "FTP Security Extensions".
- Third-party control of data transfer: To manage large datasets for distributed communities, authenticated third-party control of data transfers between storage servers must be provided. A third-party operation allows a user or application at one site to initiate, monitor and control a data transfer operation between two other sites: the source and destination for the data transfer. Our implementation adds Generic Security Services (GSS)-API authentication to the existing third-party transfer capability defined in the FTP standard.
- Parallel data transfer: On wide-area links, using multiple TCP streams in parallel (even between the same source and destination) can improve aggregate bandwidth over using a single TCP stream. GridFTP supports parallel data transfer through FTP command extensions and data channel extensions.
- Striped data transfer: Data may be striped or interleaved across multiple servers, as in a DPSS network disk cache. GridFTP includes extensions that initiate striped transfers,

which use multiple TCP streams to transfer data that is partitioned among multiple servers. Striped transfers provide further bandwidth improvements over those achieved with parallel transfers. The GridFTP protocol extensions was defined to support striped data transfers.

- Partial file transfer: Some applications can benefit from transferring portions of files rather than complete files: for example, high-energy physics analyses that require access to relatively small subsets of massive, object-oriented physics database files. The best that the standard FTP protocol allows is transfer of the remainder of a file starting at a particular offset. GridFTP provides commands to support transfers of arbitrary subsets or regions of a file.
- Automatic negotiation of TCP buffer/window sizes: Using optimal settings for TCP buffer/window sizes can dramatically improve data transfer performance. However, manually setting TCP buffer/window sizes is an error-prone process (particularly for non-experts) and is often simply not done. GridFTP extends the standard FTP command set and data channel protocol to support both manual setting and automatic negotiation of TCP buffer sizes for large files and for large sets of small files.
- Support for reliable and restartable data transfer: Reliable transfer is important for many applications that manage data. Fault recovery methods are needed to handle failures such as transient network and server outages. The FTP standard includes basic features for restarting failed transfers that are not widely implemented. GridFTP exploits these features and extends them to cover the new data channel protocol.

Advanced Grid Middleware

The components of this layer are serving more advanced mechanisms in The Grid. They can be called collective services because they operate on many lower level resources. In most cases they are not required for the environment but they increase quality of service and can serve more advanced scenarios for the Grids.

The important part of the components of this layer is taken from Gridge Toolkit. Gridge Toolkit is an internal PSNC open source software initiative aimed to help users to deploy ready-to-use grid middleware services and create productive Grid infrastructures. All Gridge Toolkit software components have been integrated together and form a consistent distributed system following the same interface specification rules, license, quality assurance and testing.

Gridge tools and services enable applications to take advantage of dynamically changing grid environment. These tools have ability to deliver dynamic or utility computing to both, the application users and developers and resource owners. Through supporting the shared, pooled and dynamically allocated resources and services, managed by automated, policy-based Grid Resource Management System (GRMS) that interfaces with such services as Mercury monitoring system, adaptive component services, data and replica management services and others, the Gridge offers the state of the art dynamic grid features to applications. Gridge technology can be used by various kinds of businesses, including vendors, but also financial companies, service organizations

etc.

- Grid Resource Management System (GRMS)

The Grid Resource Management System (GRMS) is an open source meta-scheduling system, which allows developers to build and deploy resource management systems for large scale distributed computing infrastructures. The GRMS, based on dynamic resource selection, mapping and advanced scheduling methodology, combined with feedback control architecture, deals with dynamic Grid environment and resource management challenges, e.g. load-balancing among clusters, remote job control or file staging support. Therefore, the main goal of the GRMS is to manage the whole process of remote job submission to various batch queuing systems, clusters or resources. It has been designed as an independent core component for resource management processes which can take advantage of various low-level Core Services and existing technologies. Finally, the GRMS can be considered as a robust system which provides abstraction of the complex grid infrastructure as well as a toolbox which helps to form and adapts to distributing computing environments.

The GRMS is a central point for all resource and job management activities and tightly cooperates with other services: authorization, information systems, monitoring, data management to fulfill applications requirements. The main features of the GRMS are:

- job submission
- job control (suspending, resuming, canceling)
- ability to choose “the best” resource for the job execution using multicriteria matching algorithm
- support for the job checkpointing and migration
- support for file staging
- storing all information about the job execution
- user notifications support
- workflow jobs support

The GRMS has been designed as an independent set of components for resource management processes which can take advantage of various low-level Core Services, e.g. GRAM, GridFTP and Grid Monitoring System, as well as various grid middleware services, e.g. Grid Authorization Service, Grid Data Management Service. All these services working together provide a consistent, adaptive and robust grid middleware layer which fits dynamically to many different distributing computing infrastructures. The GRMS implementation requires Globus software to be installed on grid resources, and uses Globus Core Services deployed on resources: GRAM, GridFtp, MDS (optional). The GRMS supports Grid Security Infrastructure by providing the GSI-enabled web service interface for all clients, e.g. portals or applications, and thus can be integrated with any other middleware grid environment.

One of the main assumptions for the GRMS is to perform remote jobs control and management in the way that satisfies Users (Job Owners) and their applications requirements. All users requirements are expressed within XML-based resource specification documents and sent to the GRMS as SOAP requests over GSI transport layer connections. Simultaneously, Resource Administrators (Resource Owners) have full control over resources on which all jobs and operations will be performed by appropriate GRMS setup and installation. Note, that the GRMS together with Core Services reduces operational and integration costs for Administrators by enabling grid deployment across previously incompatible cluster and resources. Technically speaking, the GRMS is a persistent service within a Tomcat/Axis container. It is written completely in Java so it can be deployed on various platforms.

GRMS supports Grid Security Infrastructure by providing GSI-enabled Web Service interfaces and in fact acts on behalf of end users. The communication between the GRMS service and all clients is done through a GSI-enabled HTTP-based protocol called httpg implementing transport-level security introduced by Globus community. With the GAS, GRMS is able to manage both, job grouping and jobs within collaborative environments according to pre-defined VO security rules and policies. With the Data Management services from Gridge, GRMS can create and move logical files/catalogs and deal with data intensive experiments. Gridge Monitoring Service can be used by GRMS as an additional information system. Finally, Mobile service can be used to send notifications via SMS/emails about events related to users' jobs and as a gateway for GRMS mobile clients.

GRMS is able to store all operations in a database. Based on this information a set of very useful statistics for both end users and administrators can be produced. All the data are also a source for further, more advanced analysis and reporting tools. All users preferences and job requirements must be expressed as XML-based resource specification documents, called GRMS Job Description. Once such a request is sent, each job within GRMS receives a unique ID and the whole process of job scheduling and control begins.

- Gridge Authorization Service (GAS)

The Gridge Authorization Service (GAS) is an authorization system which can be the standard decision point for all components of a system. Security policies for all system components are stored in GAS. Using these policies GAS can return an authorization decision upon the client request. GAS has been designed in such a way that it is easy to perform integration with external components and it is easy to manage security policies for complex systems. The possibility to integrate with the Globus Toolkit and many operating system components makes GAS an attractive solution for grid applications.

Generally, an authorization service can be used for returning an authorization decision upon the user request. The request has to be described by three attributes: user, object and operation. The requester simply asks if the specific user can perform the operation on the specific object. Obviously, the query to an authorization service can be more complex and the answer given by such service can be complicated, too. One of the services which can work in such scenario is the Gridge Authorization Service (GAS). GAS has been designed in a form which enables many possible applications. GAS can communicate in many ways with other components. By using the modular structure of GAS it is easy to write a completely new commu-

nication module. The GAS complex data structure can be used to model many abstract and real world objects and security policies for such objects. For example, GAS has been used for managing security policies: for many Virtual Organizations, for services (like Grid Resource Management Service, iGrid, Mobile Services and other) and for abstract objects like communicator conferences or computational centers. These and many other features give a possibility to integrate GAS with many existing solutions. Such integration can be very important, because it raises the security level of the existing solutions and makes it possible to use the newest security technologies.

The main goal of GAS is to provide a functionality that would be able to fulfill most authorization requirements of grid computing environments. GAS is designed as a trusted single logical point for defining security policy for complex grid infrastructures. As flexibility is the key requirement, it is to be able to implement various security scenarios, based on push or pull models, simultaneously. Secondly, GAS is considered as independent of specific technologies used at lower layers, and it should be fully usable in environments based on grid toolkits as well as other toolkits. The high level of flexibility is achieved mainly through the modular design of GAS and usage of a complex data structure which can model many scenarios and objects from the real world. It means that GAS can use many different ways for communication with external components and systems, use many security data models and hold security policy on different types of storage systems. These features make GAS attractive for many applications and solutions (not only for those related with grids). GAS has to be the trusted component of each system in which it is used and it brings about that the implementation of GAS was written in ANSI C. This choice makes GAS a very fast and stable component which uses not much CPU power and little amount of memory. The main problem of many authorization systems is their management. It is not easy to work with a complex system in a user-friendly way. Based on many experiences and the end user comments together with GAS, the GAS administration portlet (web application) is provided, which makes management as easy as possible. Flexibility of this solution gives users a full possibility of presenting only these security policies which are important for them. The GAS management is possible in two other ways: by the GUI GTK client and by the command line client.

- Grid Data Management System (GDMS)

Data storage, management and access in Grid environment is supported by the Grid Data Management Suite (DMS). This suite composed of several specialized components allows to build a distributed system of services capable of delivering mechanisms for seamless management of large amount of data. This distributed system is based on the pattern of autonomic agents using the accessible network infrastructure for mutual communication. From the external applications point of view DMS is a virtual file system keeping the data organized in a tree structure. The main units of this structure are metadirectories, which enable to put a hierarchy over other objects and metafiles. Metafiles represent a logical view of computational data regardless of their physical storage location.

Data Management System consists of three logical layers: the Data Broker, which serves as the access interface to the DMS system and implement the brokering of storage resources, the Metadata Repository that keeps information about the data managed by the system, and

the Data Container, which is responsible for the physical storage of data. In addition, DMS contains modules which extend its functionality to fulfill the enterprise requirements. These include the fully functional web based administrator interface and a Proxy to external scientific databases. The Proxy provides a SOAP interface to the external databases, such as for example those provided by SRS (Sequence Retrieval System).

The Data Broker is designed as an access point to the data resources and data management services. A simple API of the Data Broker allows to easily access the functionality of the services and the stored data. The Data Broker acts as a mediator in the flow of all requests coming from external services, analyzes them and eventually passes to the relevant module. The DMS architecture assumes that multiple instances of the Data Broker can be deployed in the same environment, thus increasing the efficiency of data access from various points in the global Grid environment structure.

The Metadata Repository is the central element of the Gridge distributed data management solution. It is responsible for all metadata operations as well as their storage and maintenance. It manages metadata connected with the data files, their physical locations and transfer protocols that could be used to obtain them, with the access rights to the stored data and with the metadescriptions of the file contents. Currently each DMS installation must contain a single instance of the Metadata Repository, which acts as a central repository of the critical information about the metacatalogue structure, user data and security policy for the whole DMS installation.

The Data Container is a service specialized towards the management of physical data locations on the storage resources. The Data Container API is designed in a way to allow easy construction and participation in the distributed data management environment of storage containers for different storage environments. The Data Containers currently available in the DMS suite include a generic file system Data Container, a relational database Data Container and a tape archiver Data Container. The data stored on the various storage resources can be accessed with one of the many available protocols including such as GASS, FTP and Grid-FTP.

The Proxy modules are services that join the functionality of the Metadata Repository allowing to list the available databanks, list their content, read the attached metadata attributes and to build and execute queries, and of the Data Container to provide the data using the selected data transfer protocol. Such Proxy container are highly customized towards the specific platform they are working with to allow building complex queries and executing operations on the found entries.

- Grid Monitoring - Mercury

The Mercury Grid Monitoring System has been developed within the GridLab project. It provides a general and extensible Grid monitoring infrastructure. Mercury Monitor is designed to satisfy specific requirements of Grid performance monitoring: it provides monitoring data represented as metrics via both pull and push model data access semantics and also supports steering by controls. It supports monitoring of Grid entities such as resources and applications in a generic, extensible and scalable way.

The Mercury Monitoring is designed to satisfy requirements of Grid performance monitoring: it provides monitoring data represented as metrics via both pull and push access semantics and also supports steering by controls. It supports monitoring of Grid entities such as resources and applications in a generic, extensible and scalable way. It is implemented in a modular way with emphasis on simplicity, efficiency, portability and low intrusiveness on the monitored system.

The aim of the Mercury Monitoring system is to support the advanced scenarios in Grid environment, such as application steering, self-tuning applications and performance analysis and prediction. To achieve this the general GGF GMA architecture is extended with actuators and controls. Actuators are analogous to sensors in the GGF GMA but instead of gathering information, they implement controls and provide a way to influence the system.

The architecture of Mercury Monitor is based on the Grid Monitoring Architecture (GMA) proposed by Global Grid Forum (GGF), and implemented in a modular way with emphasis on simplicity, efficiency, portability and low intrusiveness on the monitored system.

The input of the monitoring system consists of measurements generated by sensors. Sensors are controlled by producers that can transfer measurements to consumers when requested.

Sensors are controlled by producers that can transfer measurements to consumers when requested. Sensors are implemented as shared objects that are dynamically loaded into the producer at run-time depending on configuration and incoming requests for different measurements.

In Mercury all measurable quantities are represented as metrics. Metrics are defined by a unique name such as `host.cpu.user` which identifies the metric definition, a list of formal parameters and a data type. By providing actual values for the formal parameters a metric instance can be created representing an entity to be monitored. A measurement corresponding to a metric instance is called metric value.

Metric values contain a time-stamp and the measured data according to the data type of the metric definition. Sensor modules implement the measurement of one or more metrics. Mercury Monitor supports both event-like (i.e. an external event is needed to produce a metric value) and continuous metrics (i.e. a measurement is possible whenever a consumer requests it such as, the CPU temperature in a host).

Continuous metrics can be made event-like by requesting automatic periodic measurements. In addition to the functionality proposed in the GMA document, Mercury also supports actuators.

Actuators are analogous to sensors but instead of taking measurements of metrics they implement controls that represent interactions with either the monitored entities or the monitoring system itself. In addition to all mentioned features Mercury facilitates steering.

- Mobile User Services

Mobile software development in Gridge is focused on providing a set of applications that

would enable communication between Mobile devices, such as cell phones, Personal Digital Assistants (PDA) or laptops and Grid Services on the other side. This class of applications is represented by clients running on mobile devices, mobile gateways acting as a bridge between clients and Grid services as well as additional specialized middleware services for mobile users.

The main goal of the services is to make use of small and flexible mobile devices that are increasingly used for web access to various remote resources. The system provides Grid access mechanisms for such devices. This requires adoption of the existing access technologies like portals for low bandwidth connectivity and low level end-user hardware. The mobile nature of such devices also requires flexible session management and data synchronization. The system enhances the scope of present Grid environments to the emerging mobile domain. Utilizing new higher bandwidth mobile interconnects, very useful and previously impossible scenarios of distributed and collaborative computing can be realized. To achieve this and taking into consideration some still existing constraints of mobile devices, the Access for Mobile Users group is developing a set of applications in the client-server model with the J2ME CLDC/MIDP- java client, and portlet server working with GridSphere. This set allow us to manage end user Grid jobs (steer an application) or view messages and visualizations produced by Grid applications on device such simple as standard mobile phone. The second group of developed services is tightly connected with end user notifications about various events in Grids. Events like: the information about user application is started or finished, the visualization is ready for viewing or waiting for new data, can be send to end users using various notifications way. It can be Email, SMS, MMS, or message of one of Internet Communicators like AIM, Yahoo, ICQ, Jabber etc. (including most popular in Poland Gadu-Gadu and Tlen).

Mobile services gives also end users possibility to start a conference concerning aforementioned event between users of given virtual organization (including conferences between clients of different communicators).

The unique possibility of giving access to Grid resources for users of relatively weak devices is one of features that distinguish Gridge mobile applications from other Grid systems. Moreover, the used technology, Java 2 Micro Edition – Mobile Information Device Profile (J2ME-MIDP) applications (midlets) on the client side allows to develop flexible, possibly off-line working programs that may be used on a wide range of devices supporting J2ME. Using the MIDP compliant device internal repository for storing data, gives the user possibility to use it later in offline state and prepare the data, to be sent in on-line state.

The Mobile Command Center (MCC) that acts as a gateway between mobile client and Grid services is developed in Java as a GridSphere portlet (see Gridsphere.org) with separate "mobile" context. MCC automatically grabs the device profile (like device class, screen size, color depth, etc), this information is used during forwarding the request from mobile device to Grid services (mainly GSI-enabled Web Services like Gridge MessageBox, Visualization Service for Mobiles or Gridge Resource Management System). Services that can be accessed from mobile device using MCC belong to two groups: the first group consists of Grid services that were adopted to use with mobile devices, the second group are services developed for use only with mobile devices. The Visualization Service for Mobiles belong to second

group and is used to view the application output in form of visualization prepared exactly according to the User's device capabilities. The advantage in this case is as follows: the large amount of data is not sent via weak GPRS connections to the device that cannot store it in the memory and cannot display it correctly. First group of services consists of Gridge Resource Management System and Notification and Messenger Service. The first service can be used in 'Collaborative scenario' - the user can steer the application (even not being an owner) from mobile device. He/she can get the jobs list, migrate, resume, suspend, cancel, edit, view history and submit new job on the basis of edited/modified description of already finished jobs. Using GRMS together with Notification service the user can register for user notifications related to the running jobs. In this way the user is notified about important events occurring in the Grid (like jobs status changes, application output availability). These notifications can be send as Email, SMS and Internet Communicator (AIM, Yahoo etc) messages to the user. Using the Messenger Service it is possible also to make a conference between users of Virtual Organization defined in Gridge Authorization Service even if they use different communicators.

- OGSA-DAI

OGSA-DAI is a middleware product which allows data resources, such as relational or XML databases, to be exposed on to Grids. Various interfaces are provided and many popular database management systems are supported. The software also includes a collection of components for querying, transforming and delivering data in different ways, and a simple toolkit for developing client applications. OGSA-DAI is designed to be extensible, so users can add their own additional functionality.

The architecture of OGSA-DAI consists of a five layers each serving a different purpose. The lowest layer is Data Layer. It consists of the data resources that can be exposed via OGSA-DAI. Currently these include:

- Relational databases such as MySQL, SQL Server, DB2, Oracle, PostgreSQL,
- XML databases such as eXist, Xindice
- Files and directories in formats such as OMIM, SWISSPROT and EMBL

Business Logic Layer layer encapsulates the core functionality of OGSA-DAI. It consists of components known as data service resources. Multiple data service resources can be deployed to expose multiple data resources. There is a 1-1 relationship between data service resources and data resources. The responsibilities of a data service resource include:

- Execution of perform documents - a perform document describes the actions that a data service resource should to take on behalf of the client. Each action is known as an activity. OGSA-DAI already includes a large number of activities for performing common operations such as database queries, data transformations and data delivery.
- Generation of response documents - a response document describes the status of execution of a perform document and may contain result data, such as the results from a database query.

- Data resource access - interactions with data resources take place via the data resource accessor component.
- Data transport functionality - data can be streamed in and out of data service resources to and from clients and other data service resources.
- Session management - the creation, access and termination of session objects allowing state to be stored across multiple requests to the data service resource. All perform document requests are processed within a session. Sessions are also used for storing the streams used by the data transport functionality. These are known as session streams.
- Property management - the creation, access and removal of properties associated with the data service resource. These are known as data service resource properties and are generally used for exposing meta-data such as the status of a request or the schema of the underlying data resource.

Presentation Layer encapsulates the functionality required to expose data service resources using web service interfaces. OGSA-DAI includes two realisations, one compliant with WSRF and the other compliant with WSI (that is a solution that only relies on the specifications mentioned in the WS-I basic profile, i.e. that do not use WSRF).

A client can interact with a data service resource via a corresponding data service. Depending on whether a WSRF or WSI data service has been deployed, the client application must be compliant with the WSRF or WSI standards.

ACGT Business Processes Services

Components located on that layer should operate in terminology understandable by end users: biologists, clinicians, bioinformaticians. Their role should be: to take care about specific users requests, to translate users requests to specific operation on grid environment (computation, data retrieval)

- Semantic Mediator

The basic role of the Mediator within the ACGT environment is to provide ACGT users/tools with a powerful mechanisms for retrieving data from integrated database systems (originally distributed and heterogeneous).

During last years, advances in biomedical studies and related areas have led to an overwhelming increase in the number and size of databases in such topic. Researchers usually must analyse each of those databases in order to successfully conduct their experiments. Furthermore, each database may offer specific query interfaces, forcing the researcher to learn its details.

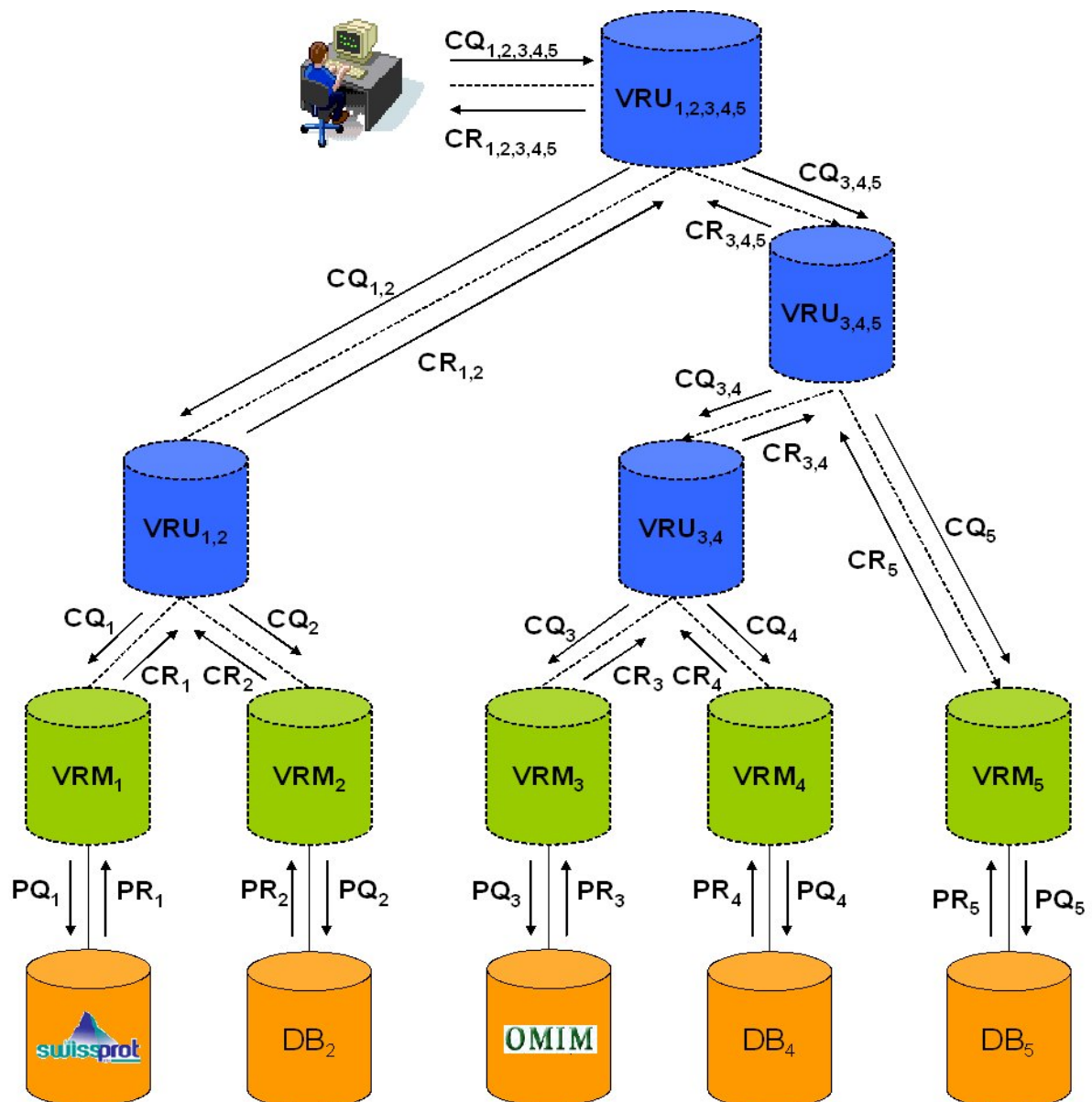


Figure 5.2. Role of the Mediator

Semantic mediation deals with this problem. It aims at offering a seamless and homogeneous to heterogeneous and distributed data sources. This accomplished by offering users a unique query interface which allows accessing several distributed databases in a transparent manner.

In order to accomplish this, the Mediator must deal with two major tasks:

- Solving schema level inconsistencies: this means that different database schemas must be unified into a single schema. For example, two columns may refer to the same concept, but be differently named.

- Solving Instance level inconsistencies: this means that the same data may be differently expressed in two databases. For example, two different identifiers could refer to the same instance of a concept, of different units could be used in the same measure.

The Mediator will provide ACGT users and tools with a data gathering from distributed and heterogeneous sources. It will also provide the necessary query interfaces and visualization models. The query interface to be offered to end users must be simple enough for clinicians to use, and admit some complex queries. Discussions are already ongoing to define the requirements for the query language to be implemented. The Mediator will offer services to KDD tools, therefore an API must be provided. There is a proposal of wrapping the tool with Web-Services, thus taking advantage of its high compatibility and ease of use. There must also be an agreement on the format of returned results. Probably, XML language will be used in this aspect.

The Mediator will be right on top of data access mechanisms. This will be in charge of providing wrappers, which will offer seamless access to different kinds of data sources. Therefore a common query language will be offered. SQL or an adaptation of it will probably be chosen.

- Master Ontology Service

The role of Master Ontology Service should be to describe the world used by end users and provide a mapping between terms and physical resources in the Grid environment. That functionality makes it one of the most important points of integration in the project

- Workflow Manager

Workflow Manager is a component responsible for execution and management of workflows defined by end user and submitted to ACGT environment.

- Knowledge Discovery Tools

The role of Knowledge Discovery (KD) Tools is to provide a set of operators to analyse clinico-genomic data, which can be connected into a KD workflow using a workflow editor. Provides operators for data visualization, transformation, modeling, and for organizing and optimizing workflow (Note: this concerns only the KID part of the workflow)

- ACGT Analytical Services

Analytical Services in ACGT environment are specialized services providing some precise functionality for end user (data processing, data transformation). The services should be developed according to some methodology and using uniform technology (eg. web services). They will need to be easily discovered and invoked. To achieve that goal interfaces should be well defined and strongly typed operating on terms defined in common meta data repository.

- VO Management Service

The service responsible for creation and management of virtual organization for ACGT con-

sortium. It should provide means for adding new users, for managing roles of users in the system, and relationships between users.

User Access Layer

It is rather difficult to point out any particular component from this layer because there will be variety of different tools and client application placed here. Some of them will be for general use, and there will be a lot specialized clients for specific tasks. One of possible generic application could be workflow editor for creation and submission complicated tasks.

6. ACGT System Vision - Example

In this section simple scenario is presented. The intention is to provide a overview how a particular components will be cooperating to provide required functionality. Presented scenario is based on one described in D2.1 deliverable: "User requirements and specification of the ACGT internal clinical trial". The scenario is called "In-silico modeling of tumor response to therapy" (scenario SC5). Example presented here is a simplified version of that scenario, showing some basic interactions between different ACGT components placed in context of introduced architecture.

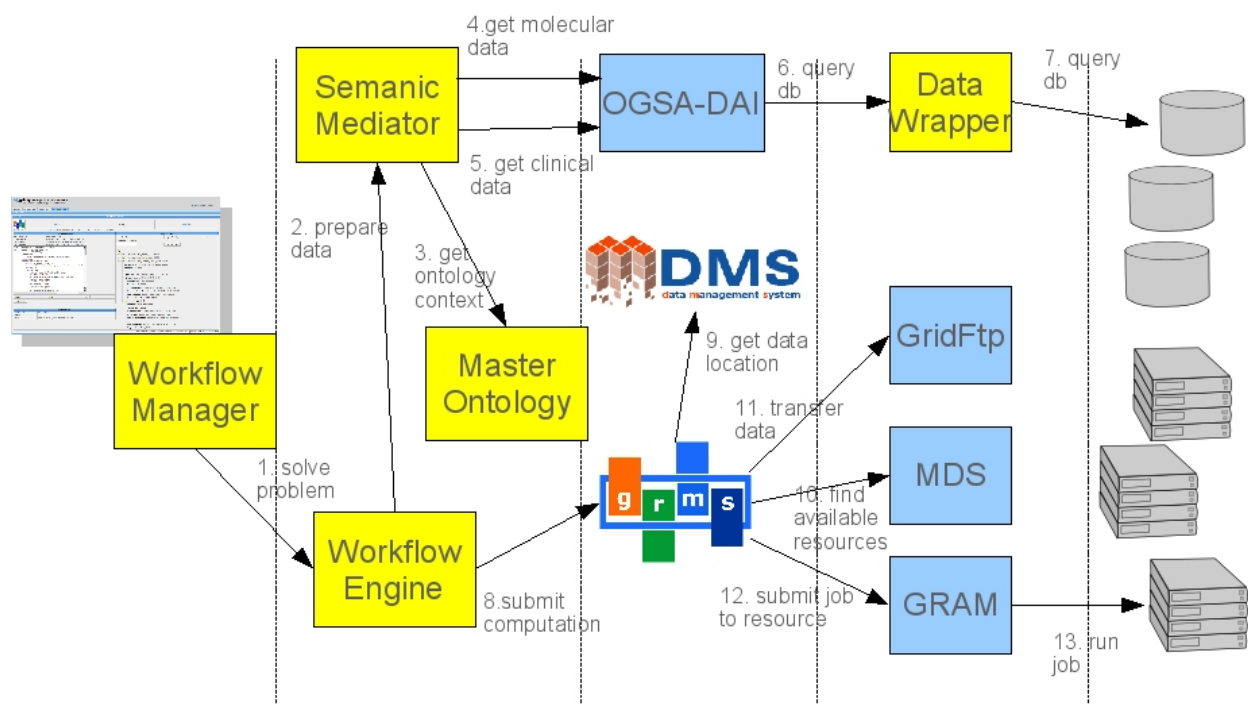


Figure 6.1. Simple scenario

The idea of the presented scenario is to do some computation on the Grid in a transparent way using different data sources for computation. At first user constructs a request to the ACGT System as a workflow of separated tasks, using client application with nice GUI. The workflow is described using vocabulary familiar to the user and defined in the ACGT Business Processes Layer. Such kind of description goes to the Workflow Engine responsible for managing the execution of the user request. The first step of the workflow is to retrieve some data that will then be used for computational job. The Workflow Engine (WE) calls the Semantic Mediator (SM) module and asks for required data. The SM using knowledge contained in the Master Ontology formulates query to the Grid Layer. Some Data Wrapper is used between physical database and a Grid Service. As soon as data is retrieved the WE prepares description for metascheduler (GRMS) to perform computation in the Grid environment. GRMS takes care of whole process of computation: chooses "the best" machine, prepares execution environment by transferring input data, and collect output data generated by job to make them available by upper layers. After the computational job is done, the WE notifies end user about the results.

The idea of the presented scenario is not to provide detailed information about interactions among particular components. The most important issue is to visualize differences between layers in architecture - level of abstraction on layers.

References

[GRIDGE] *Gridge home page*. <http://www.gridge.org/>.

[GLOBUS] *Globus Toolkit home page*. <http://www.globus.org/>.

[INTELI] *InteliGrid project home page*. <http://www.inteligrd.com/>.

[CABIG] *caBIG home page*. <https://cabig.nci.nih.gov/>.

[MYGRID] *MyGrid home page*. <http://www.mygrid.org.uk/>.

[DATAM] *DataMiningGrid home page*. <http://www.datamininggrid.org/>.